```
                                ab#)
(0B2a6              ,           b#)
(0B2a6a6            ,            #)
(0B2a6a6b7          ,            #)
(0B2a6a6b9          ,            #)
(0B2a6B9            ,            #)
(0B2B5              ,            #)
(0S1                ,            #)
    acc
```

3.3.3 Analyse LALR(1).

La méthode SLR(1) est intéressante quant au stockage de sa table d'analyse qui occupe peu de place. Malheuresement c'est une analyse qui accepte une classe restreinte de grammaire.

La méthode LR(1) vient pour pallier à cet inconvenient important, mais celle-ci nous surprend quand à l'importance de la place requise pour le stockage de sa table d'analyse.

Une méthode intermédiaire serait donc souhaitable. l'analyse LALR(1) a été conçue dans ce sens là. Sa table d'analyse occupe exactement la même place que celle de la méthode SLR(1) et aussi elle satisfait une grande classe de grammaires.

3.3.3.1 Construction De La Table D'analyse LALR(1). -

Nous la déduirons très simplement à partir de la table d'analyse LR(1) et de la collection des items LR(1). Dans un item LR(1) apparaissent deux parties: le corps (item LR(0)) et le lookahead.

Pour constituer les ensembles d'items LALR(1), nous rassemblerons les ensembles dont les items LR(1) ont respectivement les mêmes corps. Autrement dit ce qui différencient les ensembles à grouper seront les lookaheads attachés aux items LR(1).

Un item LALR(1) de l'ensemble nouvellement créé aura le même corps que ceux des items LR(1) des ensembles regroupés et

aura pour lookahead, la réunion des lookheads des items LR(1) des ensembles regroupés. Ainsi le nombre des ensembles et donc des états de l'automate diminue et égalise le nombre des états obtenus par la méthode SLR(1).

Une fois les états déterminés, nous procéderons à la construction de la table LALR(1). Cette table est tout simplement la table LR(1) mais condensée dans le sens où des lignes de la table LR(1) se supperposent pour former une ligne de la table LALR(1).

exemple

Reprenons l'exemple de l'analyse LR(1) et plus particulièrement les ensembles des items LR(1). Les états ou ensembles nouvellement créés pour l'analyse LALR(1) sont les suivants:

- I0 : [S' --->.S,#]
  [S --->.BB,#]
  [B --->.aB,a/b]
  [B --->.b,a/b]

- I1 : [S' --->S.,#]

- I2 : [S --->B.B,#]

- I3,I6 : [B --->a.B,a/b/#]
  [B --->.aB,a/b/#]
  [B --->.b,a/b/#]

- I4,I7 : [B --->b.,a/b/#]

- I5 : [S --->BB.,#]

- I8,I9 : [B --->aB.,a/b/#]

A partir de la table LR(1), pour obtenir la table LALR(1), nous superposerons les lignes 3 et 6 ensemble, les lignes 4 et 7 ensemble et les lignes 8 et 9 ensemble. Les autres lignes resteront inchangées.

En procédant à la superposition des lignes, plusieurs cas de conflits peuvent se présenter.

Etudions chacun de ces cas:

-cas 1 : présence de deux décalages différents dans une même case.

Ces deux décalages proviennent des ensembles LR(1) qui ont le même corps d'items.(Le groupage des ensembles a été fait selon ce critère). Il y aurait conflit dans ce cas, lorsque les états de transition ne seront pas identiques ou équivalents(c'est à dire groupés ensemble). Or, l'état suivant est défini comme étant la closure de transition par l'élément se trouvant après le point dans l'item (qui est le même dans les deux items LR(1)). D'après la définition de la closure, les états de transition seront identiques ou équivalents(leurs items auront respectivement le même corps)

-cas 2 : Présence d'un décalage et d'une réduction dans une même case.

Ces deux actions proviennent de deux items LR(1) (de deux états LR(1) différents) ayant les mêmes corps. Mais dans la case de la table LR(1) qui contient la réduction, on aurait eu aussi un décalage puisque le corps de l'item LR(1) provoquant cette réduction est le même que celui qui a provoqué le décalage dans la case symétrique(dans l'autre ensemble). Cette case serait donc multidéfinie dans la table LR(1). Et par conséquent la grammaire ne serait pas LR(1). Ceci a été exclu initialement. Nous pouvons donc déduire que si une grammaire n'est pas LR(1), elle ne peut être LALR(1) (voir définition plus tard).

-cas 3 : Présence de deux réductions différentes dans une même case.

Le seul conflit possible qui peut se présenter est la présence dans une même case de deux réductions différentes. En effet, deux items LR(1) ayant respectivement la forme [A --->C.,x] et [B --->C.,x] peuvent exister dans un même ensemble. Maintenant, si on essaye de superposer(grouper) deux ensembles ayant ces deux items, il y aurait conflit entre deux réductions dans une même case.(voir exercice 3.3)

-cas 4 : Présence d'une erreur et d'une autre action dans

une même case.

Le cas (erreur,décalage) ne peut pas se présenter car si on a un décalage dans une case, on aurait eu un autre décalage dans la case symétrique pour la même raison évoquée dans le cas 1.

Dans le cas (erreur réduction), si la chaine est correcte syntaxiquement, l'analyse LR(1) et l'analyse LALR(1) progresseront exactement de la même façon (voir les exemples d'analyse de chacune des deux méthodes). La seule chose qui diffère réside dans les appelations ou numérotage des états de transition.

Par contre lorsqu'il y a une erreur dans la chaine à analyser, l'analyse LR(1) détectera plus rapidement cette erreur alors que l'analyse LALR(1) procèdera à une série de réductions avant la rencontre de l'erreur.

exemple:

La table d'analyse LALR(1) de la grammaire de la figure 3.6 est la suivante:

	a	b	#	S	B
0	d36	d47		1	2
1			acc		
2	d36	d47			5
36	d36	d47			89
47	r3	r3	r3		
5			r1		
89	r2	r2	r2		

Figure 3.9: Table d'analyse LALR(1).

ANALYSE SYNTAXIQUE

Analysons la chaine aabaab#

(0	,	aabaab#)
(0a36	,	abaab#)
(0a36a36	,	baab#)
(0a36a36b47	,	aab#)
(0a36a36B89	,	aab#)
(0a36B89	,	aab#)
(0B2	,	aab#)
(0B2a36	,	ab#)
(0B2a36a36	,	b#)
(oB2a36a36b47	,	#)
(0B2a36a36B89	,	#)
(0B2a36B89	,	#)
(oB2B5	,	#)
(oS1	,	#)
acc		

Analysons maintenant la chaine aab#

(0	,	aab#)
(0a36	,	ab#)
(0a36a36	,	b#)
(0a36a36b47	,	#)
(0a36a36B89	,	#)
(0a36B89	,	#)
(0B2	,	#)
erreur		

Nous cloturons cette section en mentionnant qu'une autre méthode plus efficace pour la construction de la table LALR(1) existe. Cette méthode plus pratique, évite le biais de la construction de la table LR(1). Cependant la méthode présentée dans ce cours nous a permis de cerner les avantages et les inconvénients de chacune des méthodes SLR(1), LALR(1) et LR(1).

3.4 ANALYSE DES GRAMMAIRES AMBIGUES.

Lorsqu'une grammaire est ambigue, sa table d'analyse LR(1), SLR(1) ou LALR(1) est multidéfinie. Nous pouvons résoudre les conflits d'actions apparaissant dans la table et ceci en tenant compte du contexte du langage source. Par exemple,